200 points. Individual Work Only. Due November 30, 2024, before 11:59 PM.

Objectives:

To design and implement a GPU version of the "Game of Life" program in C/C++.

Problem Statement:

The objectives of this homework are:

1. Design and implement a GPU version of the Game of Life program.
2. Test the program for functionality and correctness. Make sure to test the output of the GPU version with the output of the CPU version of the program, the results must be identical.
3. Measure the performance of the program and optimize the program to improve performance. Evaluate the performance of GPU version with the corresponding CPU serial version, the best OMP version, the best MPI Version and include the findings in the report.
4. Review the article provided in Blackboard (Millán, E.N., Wolovick, N., Piccoli, M.F. et al. Cluster Computing (2017) 20: 2763. https://doi.org/10.1007/s10586-017-0850-3), incorporate at least one of the optimizations described in the paper, and describe the optimizations you have implemented along with the performance improvements obtained in the report.

Guidelines and Hints:

1. Review GPU programming tutorials and other resources provided in Blackboard.
2. Initialize the array using a random number generator with the same seed on the CPU and use this array to test both the CPU and GPU versions of the programs. Make sure to compare the output from both versions, they must be identical. You can use diff command in the shell script to check the difference. Try to use MPI version with 20 processors to calculate the time for CPU version.
3. Initially you can just use the GPU global memory to get started, but make sure to use the GPU shared memory to optimize your program performance (like the matrix transpose example we discussed in class).
4. Execute your program on the cluster asax.asc.edu on a single CPU and a single GPU and note down the time taken. Use the matrix size 5000x5000, 10000x10000 and maximum iterations as 5000 for both versions (CPU and GPU).
5. Make sure you submit jobs to the compute nodes and DO NOT run any jobs on the login node/head node on the ASA cluster. Also make sure you submit your jobs ONLY to the “class” queue. Use the *run_gpu* script to submit GPU jobs.
6. Comment out any print statements you might have to print the board when you execute with larger problem sizes. Also execute the program three times and use the average time taken.
7. Check-in the final version of your program and the job execution output files (.o<jobid> files) to the *github* server and make sure to share your *git* repository with the Instructor.

Program Documentation:

1. Use appropriate function name and variables names.
2. Include meaningful comments to indicate various operations performed by the program and instructions to compile and run the program.
3. Programs must include the following header information within comments:

```
/*
  Name:
  Email:
  Course:      CS      481/581
Homework #:
  Instructions to compile the program: (for example: gcc -Wall -O -o hw5 hw5.c)
Instructions to run the program: (for example: ./hw5 1000 1000) */
```

Submission:

Upload the source files and report (.doc or .docx or .pdf file) to Blackboard in the assignment submission section for this homework. Include the github URL in the comment section of the submission.

Grading Rubrics:

The following grading policy will be used for grading programming assignments:

Program Design and Implementation	60% (program with no compiler errors or logical errors with the required functionality)
Optimizations and improvements	15% (optimizations such as using GPU shared memory, or any one of the optimizations described in the paper)
Program Testing	10% (includes selecting appropriate test cases, performing the tests, comparing CPU and GPU results)
Report including Performance Analysis	10% (documentation of program design, implementation, optimizations, performance analysis, plotting, and test cases)
Source Code Formatting	5% (indentation, variable names, comments, etc.)

Instructions for creating github repository to manage your source code:

1. Create a github account at https://www.github.com using your crimson email address.
2. Create a new repository (say, *Fall2024_CS481_HW5*) on the github server. Make sure that you have created a private project (click on visibility to be private).

3. You can use the browser to either add files or upload files from your local machine. If you are using a terminal, you can execute the following commands in a terminal (assuming you have the solution as *hw5.c* in the directory *CS481/hw5* and repository name is *Fall2024_CS481_HW5*; use your github username instead of <githubid>):

   cd CS481/hw1
   git init
   git remote add origin https://github.com/<githubid>/Fall2022_CS481_HW1.git git
   add hw5.c
   git commit –m “Initial check in” git
   push -u origin master

4. After this if you make changes to the file hw5.c you can update the repository using:

   git commit -m "Appropriate commit message" git
   push origin master

5. Click on the setting tab, click on Manage Access on the left side, and then click on the green box in the middle of the screen “Invite a collaborator.” Enter instructor (mhchowdhury@crimson.ua.edu) and grader (**chgomes@crimson.ua.edu**) email addresses and click on “Add …” button.

The URL above (https://www.github.com/<githubid>/ *Fall2024_CS581_HW5*.git) will be different for you depending on your github id and repository names, you can get the URL by selecting HTTPS and clicking on the copy icon next to the URL when you click on the green “Code” button.